SOFTWARE DESIGN FOR A FATIGUE MONITORING DATA ACQUISITION SYSTEM

Charles Lynn Butler

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

SOFTWARE DESIGN FOR A FATIGUE MONITORING DATA ACQUISITION SYSTEM

by

Charles Lynn Butler

September 1976

Thesis Advisor: G. H. Lindsey

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Software Design for a Fatigue Monitoring Data Acquisition System		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; September 1976 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Charles Lynn Butler		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		12. REPORT DATE September 1976 13. NUMBER OF PAGES 74
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Naval Postgraduate School Monterey, California 93940		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Microcomputer, microprocessor, data acquisition system, fatigue, PL/M.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The software for an aircraft fatigue monitoring data acquisition system was designed and implemented on a prototype instrument to evaluate the feasibility of employing such a system in the data collection process for the determination of fatigue effects in aircraft structures.

The operating software and the associated hardware are

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601 |
(Page 1)

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

described herein; options for actual implementation in light of rapid technological developments are hypothesized and discussed. Separate software packages provide for the data collection and retrieval, including the pre-processing of data to a form which is suitable for computation. Results of test cases are presented where the software is proved by comparison of output to controlled input signals.

SOFTWARE DESIGN FOR A FATIGUE MONITORING DATA ACQUISITION SYSTEM

by

Charles Lynn Butler
Lieutenant, United States Navy
B.S., U.S. Naval Academy, 1969

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN AERONAUTICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL

1976

ABSTRACT

The software for an aircraft fatigue monitoring data acquisition system was designed and implemented on a prototype instrument to evaluate the feasibility of employing such a system in the data collection process for the determination of fatigue effects in aircraft structures.

The operating software and the associated hardware are described herein; options for actual implementation in light of rapid technological developments are hypothesized and discussed. Separate software packages provide for the data collection and retrieval, including the pre-processing of data to a form which is suitable for computation. Results of test cases are presented where the software is proved by comparison of output to controlled input signals.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

The life expectancy of an aircraft is exceedingly difficult to estimate; but the importance of an accurate estimate demands that maximum effort be exerted. The cost of inaccuracy is measured in both lives and dollars; that is to say, there are costs associated with operating an aircraft beyond its life expectancy as well as failing to operate an aircraft to the limit of its useful life.

The fatigue life of an aircraft is dictated by the critical points of the aircraft's structure, and is based on cumulative damage theory. An estimate of the cyclic flight loads, which might be encountered by the aircraft, is used to meet fatigue life specifications in aircraft design. Current in-service monitoring of structural fatigue damage is accomplished by a fleet-wide counting accelerometer program; however, under the same flight load, the actual strain experienced at a critical point will vary with aircraft weight, configuration, and flight condition. The fatigue monitoring data acquisition system was designed to gather significant strain information at the critical points of the structure, to record the events in the sequence in which they occur, and to make the data thus gathered available for further analysis.

The system which was developed is comprised of two major subsystems: the WRITE subsystem and the READ subsystem. The WRITE subsystem is tasked with the airborne collection of data. It makes use of a microprocessor, various components for signal processing, a magnetic tape recorder, and a strain gage network to monitor strain-generated

signals, identify significant events, and record the collected data. The READ subsystem is tasked with the retrieval of the data from the WRITE subsystem recorder. A desk-top calculator and cassette tape reader accomplish this task, which includes the read-back of the magnetic tape, the pre-processing of data, and the operation of a data file retrieval system.

A prototype system has been built and tested by controlling input signals and comparing them with the output which was retrieved. Consistent data correlation indicated that the system is feasible. The employment of the system, as envisioned, is discussed herein, as well as the options which undoubtedly will exist with the rapid advance of microcomputer technology.

II. THE WRITE SYSTEM

The WRITE subsystem of the fatigue life data acquisition system is dedicated to the sensing of significant strain events which have been experienced by the aircraft and the recording of those events. It includes various hardware components used to monitor, digitize, process and record signals which are generated by strain gages located at up to eight critical points of interest. The software package of the microprocessor controls the hardware components and monitors their functions. Screening, formatting, storing and recording of data are also functions performed by the software package.

A. HARDWARE COMPONENTS

1. MPS-803 microprocessor

The MPS-803 microprocessor is a microcomputer capable of handling data and performing arithmetic and logical operations. The microprocessor is physically comprised of three 4-1/2 inches by 6-1/2 inches printed circuit cards: the CPU card, the ROM/RAM card and the I/O card.

Of greatest significance on the CPU card are the crystal clock, which provides the timing for the microprocessor, and the 8008 CPU (central processing unit), which executes instructions located in the memory of the

microprocessor. The set of permissible instructions may be found in The Designer's Guide to Programmed Logic for MPS 800 Systems [Biewer, 1974]. The executable instructions are generally arithmetic and logic operations, addressing of input/output ports and accessing of memory locations.

The ROM/RAM card contains four PROMs (programmable read-only memory) and sixteen RAM chips (random access memory). Each PROM contains one page of memory storage for 256 bytes of information. Each byte is an instruction word represented by an 8-digit binary number (8 bits). The four PROMs, collectively referred to as ROM, provide a total of 1024 memory locations. The PROMs are the only non-volatile, unalterable storage in the microprocessor and are therefore the only suitable storage for the program instructions. This is virtually the only restriction on the size of the control program. The current program allows for at least one-half page expansion.

The sixteen RAM chips on the ROM/RAM card represent 2K ($1K=2^{10}$) memory locations. The RAM chips, collectively referred to as RAM, provide storage which is alterable by the CPU during the execution of the program. Consequently, the first page (256 bytes) of RAM is used for the storage of program variables, and the remaining seven pages (1792 bytes) of RAM are available for data storage. Unlike ROM, RAM storage is volatile and cannot be retrieved after an interruption of power.

The I/O card provides 28 TTL (transistor-transistor logic) compatible input/output lines. The circuits are selectable in groups of four to various combinations of input and output ports. The current configuration assigns the first eight lines to input port zero, the second eight lines to input port one, the third eight lines to output port two, and the remaining four lines to output port three.

2. A/D Signal Processing Module

The analog signals from the strain gages must be digitized in order to be analyzed and stored by the microprocessor. The signal processing module was designed and developed by Lt. W.C. Stanfield, U.S.N. [Stanfield, 1976]. The module provides for multiplexing of eight input channels by a multiplexer, which is driven by a binary up-down counter. The counter provides the channel number, and the multiplexer selects the channel. A sample and hold component is used to stabilize the voltage of the input analog signal, which allows signal digitization by the analog-to-digital converter (ADC). The resultant output of the ADC is an 8-bit binary representation of the analog input. The binary up-down counter and multiplexer are necessary only in order to allow monitoring of multiple channels.

3. Memodyne Recorder

The recorder employed is the Memodyne Model 171 magnetic tape recorder. The Model 171 is a write-only (record-only), parallel data machine. The recorder inputs each data byte in 8-bit parallel form and formats the word into a serial format suitable for recording on the tape. The Model 171 achieves a recording density of 40 bytes per inch of magnetic tape (320 bits/inch). The magnetic tape used is a digital quality, standard size cassette with a length of 300 feet. This allows the recording of up to 144,000 bytes (1,152,000 bits) on a single cassette. The nominal rated speed of recording is 100 bytes per second (800 bits/second); however, the developmental system was

consistently operated at 110-120 bytes per second without difficulty.

4. Peripheral Inputs

a. Header switch

The header switch is a two-position toggle switch used to advance the tape beyond its leader and to write identifying information at the beginning of the tape. Activating the switch advances the tape, ensuring that the tape leader does not interfere with the recording process. This also provides a blank portion at the beginning of the magnetic tape and further reduces the uncertainty involved in locating the recorded data when attempting to read it back. Deactivation of the switch causes the header information to be written on the tape, and the system enters the data collection mode.

b. Thumbwheel switches

The thumbwheel switches are a series of six, 10-position switches. The six switches allow the 6-digit aircraft bureau number to be indicated. These switches then become a source for identifying information to be included in the header.

c. Weight-on-wheels switch

The weight-on-wheels switch is a two-position switch, which indicates that the aircraft is on deck or airborne by sensing the extension of the landing gear oleo.

The switch is activated while the oleo is compressed, which indicates to the system that a removal of power may be imminent. The system responds by recording the data which are located in the volatile RAM to prevent its loss upon removal of power to the system.

5. Power requirements

A -10 VDC power source is required for the MPS-803 microprocessor, and a +5 VDC power source is required for the logic circuits of both the microprocessor and the Memodyne recorder. The recorder also requires a +12 VDC source for the motor drive circuit. The signal processing module also has unique power requirements for +15 VDC and -15 VDC. (Refer to figures 1 through 3.)

B. SOFTWARE PACKAGE

The WRITE system software package is a single program of instructions for the 8008 CPU. The total number of instructions is less than 1000 words, which are stored in the four PROMs of the MPS-803 microprocessor. The program was written in PL/M, a higher-level programming language for microprocessors, and converted to 8008 instructions by the PLM8 compiler, a resident program on the Naval Postgraduate School IBM-360 computer. The INTELLEC 8 microcomputer was used as the means for programming the PROMs and was also used as a developmental tool to simulate the MPS-803 microprocessor, since both utilize the 8008 CPU and 8008 instruction set. The software package is listed in Appendix A, and additional information on PL/M is included in Appendix B. See flowchart in figure 4.

1. Operation of the program

The first step of the program starts an initialization process which assigns the desired initial values for various variables used later in the program. Because the program operates by comparing successive signal inputs, the initialization provides for the first sampling of each channel to be stored in order to establish a basis of comparison for subsequent samplings. After initialization the system enters a repetitive process, which will continue until the removal of power.

The header switch is examined to determine its status. If it has been activated, the recorder motor drive is energized to advance the tape for approximately nine seconds, which is sufficient to ensure that the head of the recorder is positioned beyond the tape leader. Upon deactivation of the header switch, the thumbwheel switches are examined and the header information derived is recorded on the tape.

The weight-on-wheels switch is then examined. An activated switch indicating the aircraft is on deck causes a programmatic timer to be started which provides for the recording of data at intervals of 30-35 seconds in order to prevent loss of data in RAM upon removal of power. An airborne indication results in the normal continuation of the program, which provides for the recording of data only when the RAM capacity has been exceeded.

After the switches have been examined, the signal input channels are multiplexed. Each channel's signal is compared with its previous sample. If, as a result of this comparison, it is determined that no changes have occurred;

then the program continues by re-examining the header and weight-on-wheels switches and re-sampling the signal input channels repetitively until a change is identified.

The change of a signal is the first indication that an event may be strain significant. The sign of the slope of the change is determined and compared with the sign of the slope of the previous change. Any change in sign will identify a peak or a valley in the analog signal and is further indication that a strain-significant event may have occurred. Furthermore, to be significant, the value of the strain reading must be above a predetermined positive threshold or below a predetermined negative threshold. To aid in the analysis of the data, the first peak or valley following a strain-significant event is considered to be significant regardless of the threshold criterion. Consequently the threshold criterion functions to replace multiple occurrences within the threshold by a single event.

The mechanics of the software that accomplished this task is based on four 8-element vectors. The four vectors are used to keep track of four variables associated with each of the eight signal input channels. They are as follows:

X(CHANNEL): the current value of the signal on the channel designated

XLST(CHANNEL): the previous value of the signal on the channel designated

SLST(CHANNEL): the sign of the of last change of the signal on the channel designated (0 means negative slope; non-zero means positive slope)

FLAG(CHANNEL): an indicator showing the status of the last strain-significant event (0 means within the threshold; non-zero means outside the threshold)

Once a signal has been determined to be a strain-significant event, it is identified by channel number and stored in RAM. If an event fills the last storage location in RAM, the recording procedure is initiated. The seven pages of RAM containing the data words are transferred byte-by-byte to the recorder and written on the magnetic tape. The transfer takes slightly more than fifteen seconds, during which time the signal input channels are not monitored. Upon completion of the transfer, the seven pages of RAM are free to be refilled.

2. Analysis of the procedures

a. INITIALIZE

The first procedure encountered in the execution of the program is INITIALIZE. The first location in RAM which is to be used for the storage of data is identified as the hexadecimal address 0900H and the timer is set to zero. (Refer to Appendix A.)

```
RAM$LOC=0900H;
TIMER=0;
```

The MUX procedure is then used to fill the X(CHANNEL) vector with the first sample of the incoming signal, which is transferred to the XLST(CHANNEL) vector. The SLST(CHANNEL) and FLAG(CHANNEL) vectors are set to zero and initialization is complete.

b. MUX

The MUX procedure is used to multiplex the eight signal input channels and fill the X(CHANNEL) vector with

current values. The first command is an output instruction which causes voltages to be applied to the output port selected in the pattern which is indicated. (See Appendix B.) The instructions

```
OUTPUT(2)=40H;
OUTPUT(2)=0;
```

cause output port 2 line 7 to receive a HIGH pulse. A HIGH pulse on that line is the signal which causes the binary up-down counter to reset and results in the multiplexer selecting channel 0. Then for channels 0 through 7 the procedure which follows is performed. A pulse is created on output port 2 line 1 indicating to the analog-to-digital converter to commence conversion.

```
OUTPUT(2)=1;
OUTPUT(2)=0;
```

The next software device is used in several places in the program to wait for a particular input to the microprocessor. The statements

```
DO WHILE NOT(ROL(INPUT(1));
END;
```

cause the system to wait until an end-of-convert signal is received from the ADC, which indicates that the result of the conversion is available and valid. The value of the conversion is received on input port zero.

```
K=INPUT(0);
```

The binary up-down counter increment line is pulsed causing the next channel to be selected.

```
OUTPUT(2)=80H;
OUTPUT(2)=0;
```

Attention is then returned to processing the signal which was just received. The resolution of the ADC exceeds the requirements for an adequate strain monitoring system; consequently the 8-bit signal can be reduced to the five most significant bits with the other three bits to be used later for channel identification. The next three statements are used to perform a rounding off of the 8-bit word by

adding the bit in the third position from the right to the bit in the fourth position.

```
L=K AND 04H;
L=SHL(L,1);
K=K+L
```

For example, 1000$0101B would be rounded up to 1000$1000B, while 1000$0011B would be rounded down to 1000$0000B. The five most significant bits are then shifted to the right and stored in the X vector.

c. STORE

If an event is determined by the main program to be strain significant, the STORE procedure is called. First the data word is assembled by placing the binary representation of the number of the channel from which the signal was received in the upper three bits of the data word (LASTX). The data word already contains the signal value in the lower five bits. It is then inverted bit-by-bit (which will later facilitate reading) and stored in the RAM location which is currently indicated by the value of RAM$LOC. The variable name EVENT has been assigned to the current location.

```
EVENT=NOT(LASTX OR ROR(CHANNEL,3));
```

The FLAG(CHANNEL) is then set to indicate if the event was within or without the threshold limits according to the convention previously described.

```
FLAG(CHANNEL)=((LASTX>POS$THRESH)
    OR (LASTX<NEW$THRESH));
```

The RAM location is then incremented.

```
RAM$LOC=RAM$LOC+1;
```

If the RAM capacity has been exceeded then the RECORD procedure is invoked prior to returning.

```
IF RAM$LOC>=1000H THEN CALL RECORD;
```

d. RECORD

The number of data bytes to be recorded is determined from the RAM location indicator.

```
COUNT=RAM$LOC-1;
```

The BYTE$RECORD procedure is invoked the correct number of times to record each data byte.

```
DO RAM$LOC=0900H TO COUNT;
CALL BYTE$RECORD;
END;
```

A small blank portion of tape terminates the block of data bytes and is created by invoking the GAP procedure.

```
CALL GAP(80H);
```

Before terminating the RECORD procedure, the RAM location indicator is set to 08FFH.

```
RAM$LOC=08FFH;
```

This is an indication to the main program that an initialization process will be necessary.

e. BYTE$RECORD

Each time that the BYTE$RECORD procedure is invoked, a single data byte is recorded on the magnetic tape. First the data byte is latched in output port two, which is connected to the data input lines of the Memodyne recorder.

```
OUTPUT(2)=EVENT;
```

A pulse on line 4 of output port three indicates to the recorder that the data byte on its input lines is ready to be recorded.

```
OUTPUT(3)=08H;
OUTPUT(3)=0;
```

A time delay of approximately seven milliseconds is invoked

before the procedure is terminated in order to allow the recorder to complete the recording process.

```
CALL DELAY(77H);
```

f. GAP

The GAP procedure is used to write blank segments on the magnetic tape. The placing of line 3 of output port three in a HIGH status

```
OUTPUT(3)=04H;
```

causes the Memodyne recorder to LOAD FORWARD, a condition in which the recorder motor drive is activated but data is not written on the tape. A time delay is invoked

```
CALL DELAY(N);
```

where N is replaced by the value in the CALL GAP (N) statement. Then the LOAD FORWARD command is removed.

```
OUTPUT(3)=0;
```

g. DELAY

Only one executable statement is used in the DELAY procedure.

```
CALL TIME(VAL);
```

TIME is not a true procedure but rather a pre-defined function of PL/M which provides for a time delay. The DELAY procedure prevents multiple use of the TIME function and has the seemingly paradoxical result of reducing the number of instruction steps in the machine language program.

h. HEADER

The activation of the header switch causes the HEADER procedure to be invoked. Multiple calls of the GAP

procedure cause the tape to be advanced for approximately nine seconds.

```
DO K=1 TO 3;
DO J=1 TO 150;
CALL GAP(255);
END;
END;
```

The program enters a waiting condition which is terminated when the header switch is deactivated.

```
DO WHILE ROL(INPUT(1),3);
END;
```

Then the thumbwheel select variable C is initialized to zero in preparation for use in the READ procedure.

```
C=0;
```

The information to be written in the header is eight data bytes. Currently the first five bytes are not used. Each is programmatically filled with 80H and is available for future implementation.

```
DO RAM$LOC=0900H TO 0904H;
EVENT=80H;
END;
```

The remaining three bytes of the header are filled with the six digits of the aircraft bureau number--each half of a byte containing the binary representation of a decimal digit. The READ procedure is used to singly interrogate each one of the six thumbwheel switches and, unlike the other procedures, returns a value each time it is invoked. Because it returns a BYTE value, the procedure name, READ, is used as though it were a BYTE variable (in contrast with the conventional procedure CALL statement) with each appearance of the name READ being replaced by the value returned.

```
DO RAM$LOC=0905H TO 0907H;
EVENT=NOT(SHL(READ,4)+READ);
END;
```

The RAM location indicator is then adjusted properly for invoking the RECORD procedure, and the call is made. The eight data bytes of the header are recorded on the tape and the RAM location indicator is reset in preparation for data collection.

```
RAM$LOC=0908H;
CALL RECORD;
RAM$LOC=0900H;
```

i. READ

Each time that the READ procedure is invoked a thumbwheel switch is selected and the position of the thumbwheel switch setting is returned as a BYTE value. The first execution of READ is with C=0, which results in the thumbwheel switch corresponding to zero of select lines 2, 3 and 4 of output port two.

```
OUTPUT(2)=C;
```

The variable C is then incremented such that the next output of C will result in the bit pattern 001 appearing on lines 4,3 and 2 respectively.

```
C=C+2;
```

Then the thumbwheel switch setting is obtained from input port one, lines 1 through 4, and returned. This value is the inverted binary representation of the decimal digit indicated on the face of the thumbwheel switch.

III. THE READ SYSTEM

The READ subsystem of the fatigue life data acquisition system is dedicated to the reading (or playback) of the magnetic tape created by the WRITE subsystem. Its functions include reading, re-formatting, sorting, filing and file retrieval of the data on the magnetic tape. These tasks are accomplished with a minimum of hardware complexity and lengthy, but relatively uncomplicated, software.

A. HARDWARE COMPONENTS

1. Memodyne Recorder

The recorder employed is the Memodyne Model 172 magnetic tape recorder. The Model 172 is a read-only (playback-only), parallel data machine. The recorder reads the data in eight bit segments from the tape and assembles the word into an 8-bit parallel format. The nominal rated speed of reading is 80 bytes per second (640 bits/second); however, the developmental system was operated at approximately 25 bytes per second.

2. The HP 9830 Calculator

The Hewlett-Packard Model 9830 Calculator is the main computing component of the READ system. The HP 9830 calculator is small enough to be used on a desk top and may

be configured to accept electrical power from various sources including 110 VAC or 220 VAC. The HP 9830 is programmable from its typewriter-like console or from magnetic tape files. The magnetic tape file system is a built-in read/write cassette recorder, which allows storage of program files and data files on a digital quality, standard size cassette. The proprietary nature of the file operating system prevents the use of the built-in recorder for reading of cassette files that have been created by machines other than the HP 9800 series.

The HP 9830 calculator may be equipped with various peripheral devices. These devices include, among others, printers, x-y plotters, card readers, teletypewriters, paper tape readers and external cassette tape recorders. The use of most peripheral devices requires installation of the Extended Input/Output Read-Only Memory, which is a user-accessible plug-in device. Utilization of the software package on the HP 9830 requires 4K of augmented memory and the Matrix Operations Read-Only Memory.

3. The HP 11202A TTL I/O Parallel Interface

The Hewlett-Packard TTL I/O Parallel Interface is also required to allow the transfer of data between the HP 9830 and a peripheral device in an 8-bit parallel format. The parallel interface is a plug-in module which physically connects the I/O lines of the HP 9830 and, in this application, the I/O lines of the Memodyne recorder. The successful mating of the HP 11202A TTL I/O Parallel Interface to the Memodyne recorder was primarily due to the efforts of Lcdr. John R. Plunkett, U.S.N.

B. THE SOFTWARE PACKAGE

The READ system software package consists of two separate programs: a read/store program and a file retrieval program. Both programs are written in BASIC, a programming language which is directly executable on the HP 9830 without the requirement for external cross-compilation on a larger computer. The read/store program reads data from the external Memodyne cassette recorder and creates data files on the HP 9830 built-in cassette recorder. The file retrieval program provides for selective retrieval of data files from the HP cassette for data display or manipulation. The software package is listed in Appendix C, and additional information on BASIC is included in Appendix D. See flowcharts in figures 5 and 6.

1. The READ/STORE Program

a. Operation of the program

After the program has been loaded from the HP cassette file, the user may start the program by pressing RUN and EXECUTE. The initialization process is used to dimension the variable arrays and assign initial values.

The user is then instructed via the line printer to prepare the Memodyne recorder by holding the STOP key depressed until the clear leader on the Memodyne cassette has been advanced beyond the photoelectric sensing device of the recorder. Failure to do so will prevent the programmatic operation of the recorder. The user is

additionally instructed to continue program execution by pressing CONT and EXECUTE.

Upon continuation of the program, the header information is read from the data tape, translated and printed on the line printer. Following initialization of variables, the user is requested to input the number of pages of data to be read. A page is defined to be 256 data bytes, which corresponds to the size of a page of memory. Then, in page increments, the data is read from the data tape, translated, sorted by channel and stored in the HP 9830 memory. When 100 events corresponding to a single channel have been assimilated, the events are recorded on the HP cassette, thus freeing 100 memory locations. At the completion of the translation of the last page, all events are recorded by channel on the HP cassette even though each channel contains less than 100 events.

b. Header

The first step in translating the header information is the inputting of additional identifying information. As an example of the type of identifying information which could be used, the entering of the date, in numerical form, is requested of the user.

```
DISP "ENTER DA,MO,YR",SPA20;
INPUT N1,N2,N3
```

Then the first eight bytes on the data tape are read into the Z vector which is used for storage of unprocessed data words.

```
FOR N=1 TO 8
Z(N)=RBYTE7
NEXT N
```

The RBYTE7 command is the instruction which causes the HP 11202A parallel interface to obtain a single byte of data

from the Memodyne recorder. The interface initiates the read process by pulsing its CONTROL line, which is connected to the START line of the Memodyne recorder. The interface then monitors its FLAG line which receives the signal that the Memodyne recorder has read a byte of data and has latched it on its output lines. The data byte is then received by the calculator and stored in memory.

The first five bytes of the header information have not been used by the WRITE system and are ignored by the READ system. The remaining three bytes contain the binary-coded decimal representation of the six digits of the aircraft bureau number. The translation of these three bytes requires considerable bit manipulation. The first of these bytes corresponds to N=6. The statements

```
M1=BIAND(Z(N),240)
M1=ROT(M1,4)
```

causes the upper four bits to be converted to a decimal digit. Multiplying the digit by ten effectively places it in the tens position, and then it is added to the bureau number which is being constructed.

```
M1=10*M1
M0=M0+M1
```

M0 is initially zero. The lower four bits are similarly translated, but remain in the ones position, and are added to the bureau number.

```
M1=BIAND(Z(N),15)
M0=M0+1
```

The bureau number is now a two-digit decimal number. The next data byte is identically processed to provide the next two digits but first it is necessary to multiply the current two-digit bureau number by 100, which effectively shifts the digits to the left by two positions.

```
M0=100*M0
```

After the next two digits have been added, but before the

translation of the last data byte, the bureau number is again shifted left by two positions allowing the remaining two digits to be added to the bureau number; then the bureau number and additional header information is printed on the line printer.

```
PRINT "BUNO=";MO,"DA/MO/YR=";N1;N2;N3
```

c. Data reading and translation

After the re-initialization of variables and the number of pages of data to be read has been entered, successive pages of data are read from the data tape, translated and stored.

```
FOR I=1 TO 256
Z(N)=RBYTE7
NEXT I
```

The translation process isolates the three uppermost bits which contain the channel number.

```
J=BIAND(Z(I),224)
J=ROT(J,5)
```

Because of the use of PL/M by the WRITE system and the use of the binary up-down counter for selection of channels, it was convenient to number the eight signal input channels from zero through seven. BASIC, on the other hand, is more practically employed if the channels are lettered A through H and numbered correspondingly one through eight. The converting of the translated channel number from one scale to the other is trivial but necessary.

```
J=J+1
```

The value of the data word is found by isolating the five lowermost bits.

```
V=BIAND(Z(I),31)
```

The use of PL/M precluded the effective use of signed

arithmetic in the WRITE system. However, BASIC allows the use of negative numbers with greater ease, which prompts a re-scaling of the strain values. The WRITE system uses a scale with 0 and 32 as extrema, where 16 corresponds to the unstrained state. For the READ system it is convenient to shift the scale such that the unstrained state corresponds to zero, and the extrema are -16 and +16. Again the actual conversion is trivial.

```
V=V-16
```

Each value is stored by a subroutine which corresponds to the channel number.

```
GOSUB J OF 4100,4200,. . .,4800
```

d. Data storage

Each of the eight subroutines, corresponding to the eight channels, performs a similar storage task. Associated with each subroutine is a vector of 100 elements and an index which identifies the element. The vector names are A through H, and the indices are N1 through N8 respectively. The first subroutine will be used as an example. Each time a value is to be stored in the A vector, the subroutine at line 4100 is invoked. The value is stored at the current element location, and the index is incremented in preparation for the next element to be entered.

```
A[N1]=V
N1=N1+1
```

If the vector capacity has not been exceeded; then the subroutine task is complete.

```
IF N1≠101 THEN 4199
.
.
.
4199 RETURN
```

If the vector capacity has been reached then it is necessary to perform a preliminary task which results in the assignment of a file number for the recording of the vector on the HP cassette. This is accomplished by a subroutine.

```
GOSUB 7000
```

After a file number has been determined the STORE DATA statement may be executed.

```
STORE DATA M1,A
```

This statement causes the elements of vector A to be stored on the HP cassette in a data-type file, whose number corresponds to the variable M1.

The index and vector are then re-initialized to allow refilling of the vector.

```
N1=1
MAT A=ZER
```

The data storage task is then complete, and the program returns to translate the next data word.

e. Data file system

The data file system consists of a series of consecutively numbered files, each file being 400 bytes in length and containing a 100-element vector. The seemingly excessive file length is required for data-type files. The files are filled in numerical order as each vector in the main program reaches its storage capacity. An additional vector, L, serves as a directory, which is a list of channel numbers corresponding to the sequence in which the data files were created. The directory vector is indexed by the variable N.

The variable M0 is used to identify the file number corresponding to the first data file which is

reserved for the directory. The variable M1 is the file number of the current file. Since files 0 and 1 contain the READ system software, file 2 is the first available data file. Consequently M0 and M1 are initialized with the value 2.

```
M0=M1=2
```

Each time a vector is to be stored, the subroutine beginning at line 7000 is invoked prior to storing the vector. The channel number of the vector to be stored is entered in the directory according to the current index, and the index is incremented in preparation for the next directory entry.

```
L[N]=J
N=N+1
```

The current file number is incremented to the next available file.

```
M1=M1+1
```

With the directory entry made and the current file identified, the program continues execution with the storage of the appropriate vector.

After the last page of data has been read, translated and stored, it is necessary to file the partially filled vectors. This is done by artificially saturating the vectors and sequentially invoking each storage subroutine an additional time.

```
V=100
N1=N2=N3=N4=N5=N6=N7=N8=100
FOR J=1 TO 8
GOSUB J OF 4100,4200,. . .,4800
NEXT J
```

Finally, the directory vector must be stored in the data file which has been reserved for that purpose.

```
STORE DATA M0,L
```

2. File Retrieval Program

The file retrieval program allows the user to selectively retrieve and examine the data files which have been created by the READ/STORE program.

After the program has been loaded, the user may start the program by pressing RUN and EXECUTE. The initialization process allocates storage for two 100-element vectors: L, the directory vector, and X, the vector to be examined. The variable M is also initialized with the file number of the data file containing the directory. Then the directory file is loaded from the HP cassette into the memory storage which has been allocated for the L vector.

```
DIM L[100], X[100]
M=2
LOAD DATA M,L
```

The user is requested to designate a channel number in the range from one to eight. The channel number is echoed back on the line printer.

```
DISP "ENTER CHANNEL NO.(1-8)";
INPUT C
PRINT "CHANNEL";C,LIN1
```

Then the directory is sequentially searched for entries corresponding to the channel number which has been selected. Each occurrence of the channel number in the directory causes a subroutine to be invoked. The directory elements following the last valid entry will each be equal to zero. Consequently, encountering a zero entry in the directory indicates that the directory search is complete with respect to all valid entries. The directory search is subsequently terminated and the program returns to request that another channel number be entered.

```
      FOR N=1 TO 100
      IF L[N]≠C THEN 90
      GOSUB 120
   90 IF L[N]=0 THEN 35
      NEXT N
```

The subroutine beginning at line 120 is an example of the method used to access specific data files. The file number of the data vector to be examined is calculated by the location of the channel number (N) in the directory and the offset (M) due to the location of the directory file. The data file is then loaded into the available vector storage.

```
      LOAD DATA N+M,X
```

In this example, the file is simply printed on the line printer in a 10 by 10 array before returning to execution of the main program.

```
      FOR I=1 TO 91 STEP 10
      PRINT X(I);X(I+1);X(I+2);. . .;X(I+9)
      NEXT I
```

IV. TEST RESULTS

The testing of the fatigue monitoring data acquisition system was conducted in a laboratory environment at the Naval Postgraduate School. Channel 1 of the signal input channels was connected to ground, and the remaining 7 channels were connected to known signals. Channels 2, 3, and 4 received a sinusoidal signal with amplitude of ±3.7 volts. Channels 5, 6, 7, and 8 received an oscillating signal of variable amplitude between ±5 volts. The rate of occurrence of significant events was controlled by selecting the frequency of oscillation. The measurement of the elapsed time between input of the data and transfer to the magnetic tape was consistent with the known occurrence rate of the peaks and valleys of the input signal.

The reading of the data tape further confirmed the operation of the fatigue monitoring data acquisition system. Channel 1 was determined to be void of strain-significant occurrences. Channels 2, 3, and 4 exhibited an alternating signal of +12 and -12 which corresponds to approximately ±3.75 volts. Channels 5, 6, 7, and 8 exhibited data which was consistent with the general oscillatory signal on the corresponding signal input lines.

The analog input signal on channel 2 was 3.7 volts, and the fatigue monitoring data acquisition system processed the signal resulting in the value 12. The ADC outputs digital values on a scale 0000$0000B to 1111$1111B corresponding to input voltages in the range of ±5 volts, with 1000$0000B corresponding to zero volts. Since 3.7 volts is .74 of 5 volts, the ADC should output the binary value corresponding

to .74 of the digital scale above 1000$0000B. That value is 1101$1111B. The rounding off by the WRITE system software results in the value 1110$0000B. Then the value is shifted right three bit positions yielding 0001$1100B. The READ/STORE program interprets this value as its decimal equivalent 28. This value is re-scaled to place zero in the center, corresponding to zero strain, by subtracting 16, which results in the final value of 12. The value 12 is representative of .75 (12/16) of 5 volts, which is 3.75 volts. The values 11 and 13 would represent 3.44 volts and 4.06 volts respectively; so 12 actually represents 3.75±.15 volts, which is consistent with the analog value of 3.7 volts. (Refer to figures 7 and 8.)

V. OBSERVATIONS

A. MICROVOX RECORDER VERSUS MEMODYNE RECORDER

A successful WRITE system was designed and developed by Lt. Wesley C. Stanfield utilizing a Microvox Corporation digital wafer tape recorder and the MPS-803 microprocessor. The Microvox recorder is considerably smaller than the Memodyne recorder, and its advantages are described in "Microprogrammable Integrated Data Acquisition System: Fatigue Life Data Application" [Stanfield, 1976].

The attempted development of a Microvox READ system met with only limited success. After considerable effort and time, a Microvox read-only recorder was mated to the INTELLEC 8 microcomputer. The operation of the Microvox reader appeared to be satisfactory enough to identify some problem areas between the READ system and the WRITE system. The Microvox digital wafer is a continuous tape, which is similar in design to tape cartridges. The nature of a continuous tape allows the obliteration of previously recorded data if the tape is run beyond one full cycle. This necessitates careful identification of the beginning and end of tape. Attempts to modify the Microvox WRITE system to achieve compatibility with the requirements of the READ system resulted in repeated difficulty in obtaining consistent, reliable operation primarily due to the problems encountered in the interfacing of the various hardware components. Output signals were consistently weak making sensing of beginning of tape and end of tape unreliable and

intermittent. Use of the Microvox recorder was ultimately terminated by mechanical failure of the write-only recorder. The failure was the separation of the motor drive belt without which the tape drive capstan cannot rotate. The motor drive belt from the read-only recorder was also found to be worn. At this stage it seemed impractical to continue trying to implement the Microvox system.

After attempting to use the Microvox recorder, the Memodyne cassette recorder was once again utilized for incorporation into the fatigue monitoring data acquisition system because it was immediately available and had already been mated to the INTELLEC 8 in both the read and write modes. The Memodyne recorder had already been flight tested in a light aircraft (CESSNA 310) in a less demanding data acquisition application. In short, the reliability of the Memodyne recorder was considered to be less questionable than that of the Microvox recorder.

The Memodyne recorder cassette has 3-1/2 times the capacity of the Microvox recorder and its associated digital wafer. The Memodyne recorder is slower than the Microvox recorder, but the difference is virtually eliminated due to the requirement of the microprocessor to serially format each data word for transfer to the Microvox recorder. The Memodyne recorder, on the other hand, receives data words in 8-bit parallel format and performs its own serial formatting. The Memodyne recorder does not utilize a continuous tape requiring location of beginning of tape and end of tape, which contributes to some degree of decreased complexity but also contributes to increased weight and size. The sophistication of the Memodyne recorder's formatting system is also a contributor to increased size and weight in comparison with the Microvox recorder. Even at this the Memodyne recorder is 3-3/4 inches by 4-1/2 inches by 7 inches and weighs 4-1/2 pounds.

B. THE 8080 CPU VERSUS THE 8008 CPU

The MPS-803 microprocessor is capable of being expanded to include another 1K of ROM and an additional 2K of RAM; however the capabilities of the 8008 CPU make it possible to address a memory as large as 16K. Expansion of the system to the limits of the 8008 CPU's capabilities would increase the storage available for data collection by over eight times the capacity of the current system. The 8080 CPU can increase the addressability of memory to 64K. Over 35 times the current memory capacity for data collection would be available.

The advantage of increasing the memory capacity is that if the solid-state memory is large enough, the need for an airborne magnetic tape recording device is eliminated. The volatile nature of random access memory is still a consideration, for it would require continuous power; however technological advances may soon relieve this requirement.

C. RECOMMENDATIONS

1. Memory

The objective of the current WRITE system is to provide for multiple flights between ground servicing. If the magnetic tape recorder could be replaced by solid-state memory, a benefit could possibly be realized in weight savings but, more important, reliability could be increased.

The operation of the tape recorder in extreme temperatures and under adverse G loadings is suspect. A solid-state memory would be able to function in a much greater range of environmental conditions.

It is believed that it is within present-day capabilities to develop a system with a random access memory that can be battery-powered for a limited period of time. A non-volatile RAM manufactured by the Monolith Systems Corporation , according to its specifications, would be able to retain its contents for up to five months without recharging its batteries. A 16K memory of this type would weigh approximately six pounds with batteries and cost $2400. The cost of such a system is estimated to be twice the cost of the current WRITE system, which is approximately $1600 for component costs.

Other non-volatile, random access memory systems are currently under development. Until a mass-marketed, non-volatile random access memory is available, the most practical solution seems to be the expansion of memory to a size great enough to permit the collection of data for a single flight without exceeding the capacity of the RAM. The cassette tape recorder would continue to be an integral part of the system, but operation could be limited to the weight-on-wheels condition.

A reasonable, if not conservative, way to estimate the number of strain significant events, which might be expected over a specified length of time, is to refer to the structural design specifications for military aircraft. Table II of MIL-A-8866(ASG) specifies the frequency of manuever loads, which an aircraft must be able to sustain. An analysis of the most demanding spectrum, that of fighter aircraft with respect to strain-significant events, reveals that no more than 26,000 occurrences per 1000 flight hours

should be experienced. Based on the quarterly report on the "Aircraft Structural Appraisal of Fatigue Effects (SAFE) Program," Naval Air Development Center, Warminster, Pennsylvania, average aircraft usage rates for various models of A-6, A-7, and F-4 aircraft indicate a mean usage rate of approximately 25 flight hours per month. This implies that, statistically, an aircraft might be expected to experience 650 strain-significant events each month. Under these circumstances a fatigue monitoring data acquisition system with eight channels would have a storage requirement for 5200 data words (slightly more than 5K).

It must be acknowledged that determining storage requirements by this method is crude and imprecise, but establishing the validity of a statistical model of the time distribution of strain-significant events would most likely be very difficult. To achieve some confidence that enough storage is available to allow for deviations from the mean values cited, it is estimated that three times the mean requirement is needed. The current WRITE system requires 1.25K for program and variable storage, which would leave 14.75K available for data collection if an 8008 CPU were used to its maximum capacity. This would be marginally adequate at best, whereas if the 8080 CPU were used 62.75K fatigue significant events could be recorded. Either system would experience a similar weight and volume growth with the expansion of memory. This weight and volume growth would be approximately linear with incremental additions of 4K of memory.

Two options, which should be considered, involve the alteration of the criterion used to determine the service interval of the system. If the service interval were based on a flight-hour cycle rather than a calendar cycle, the sensitivity of storage requirements with respect to aircraft usage rates could be eliminated, and the cycle could be more

readily tailored to individual aircraft models and missions. The sensitivity of storage requirements with respect to the frequency of occurrence of strain-significant events could be eliminated if the available storage were directly monitored. For example, in the current WRITE system, the cassette tape could be visually inspected as a part of the daily inspection and replaced as required. In the case of a completely solid-state RAM storage system, an external indicator could be displayed when the memory has exceeded some predetermined saturation level. The level could be determined to allow daily inspection as well as sufficient leeway to permit subsequent flights before servicing of the system is absolutely required.

2. Recorder

The Memodyne recorder is preferred over the Microvox recorder for reasons which have been previously discussed. The Model 171 write-only, parallel data recorder is recommended for the WRITE system, and the Model 172 read-only, parallel data recorder is recommended for the READ system. The parallel data input/output is a very significant feature, which reduces software requirements in the WRITE system and greatly reduces both hardware and software requirements in the READ system. Other models of Memodyne recorders are available, which are capable of increased storage capacity but at the sacrifice of speed. The 100 series recorders have adequate tape storage capacity. Because the data collection process must be interrupted to perform the data transfer from RAM to the magnetic tape, any increased capability that sacrifices speed must be considered detrimental to the overall system.

3. Employment of the WRITE system

The system is designed to function between calendar inspections without any routine servicing required. An unused cassette is placed in the recorder, the tape having been fully rewound to the beginning leader. The power to the system is turned on, and the header switch is placed in the UP position. The tape motor will then advance the tape for a period of 7 to 10 seconds and stop. The thumbwheel switches may be checked against the aircraft bureau number while the tape is advancing. Only after the thumbwheel switches have been set should the header switch be placed in the DOWN position. Placing of the switch in the DOWN position causes the almost indiscernable movement of the tape, which is the recording of the header information. The header switch may be placed in the DOWN position before the tape has stopped advancing, but power must not be removed from the system until all movement of the tape has ceased. The header switch should be guarded and shearwired in the DOWN position to allow normal operation. At the next calendar inspection the cassette should be replaced by following the same procedure.

4. Employment of the READ system

The HP cassette with the READ system software is inserted in the cassette compartment of the HP 9830. The data tape, which has been created by the WRITE system is loaded into the Memodyne read-only recorder. The READ/STORE program is currently on file number 1 and is loaded from the HP cassette to the HP 9830 memory by pressing LOAD1 and EXECUTE. If the data files have not been previously

formatted, it is necessary to do so by entering FIND2, waiting for the tape to stop, and then entering MARK100,400. The calculator will commence marking file boundaries on the tape until it reaches the end of tape. The HP 9830 will signal an error condition which indicates that it is unable to mark more files. Then press REWIND, RUN, and EXECUTE. The instructions to advance the data tape beyond its leader will appear on the line printer. Hold the STOP key depressed, which will cause the data tape to advance. Release the STOP key when the last portion of the clear leader has advanced beyond the photoelectric sensing device, which is visible through a small hole to the right of the read head. For normal operation press CONT and EXECUTE. If no header is to be read, enter CONT1000 and EXECUTE. This is necessary when the portion of the tape with the header has already been read, and the data which follows does not have a header. When requested, enter the day, month and year in numerals separated by commas; then enter the number of 256-byte pages to be read. If a tape is to be read in its entirety, it is recommended that 25 pages be read. If there are less than 25 pages of data available, the data tape will reach its end and stop. The user must then press STOP, enter CONT2000, and press EXECUTE. If there are more than 25 pages of data available, the program will reach a normal termination. To continue reading from the same data tape, an unused HP cassette must be placed in the HP 9830. The tape must be marked by entering MARK100,400 as before. When the files have been marked, press REWIND, RUN, and EXECUTE. This time the tape will not require the leader to be advanced, nor will the reading of the header be required. The program is continued by entering CONT1000 and pressing EXECUTE.

The File Retrieval program, located on file number zero, is loaded by entering LOAD0 and pressing EXECUTE. Program execution is commenced by pressing RUN and EXECUTE.

The program is continuous and requires the user to specify the channel to be examined.

5. Data file systems

The current cassette data file system is adequate, but extremely slow in comparison to disk file systems. The HP 9830 may be equipped with a peripheral disk storage. It also may be feasible to tie the HP 9830 to a time-sharing system of a larger computer. The HP 9830 would serve as an intelligent terminal which would pre-process the input data and utilize the file system of the host computer. This would allow the retrieval and analysis of the data on the larger computer; thus circumventing the disadvantages of the HP 9830 file system and taking full advantage of the larger computer's speed and storage capability.

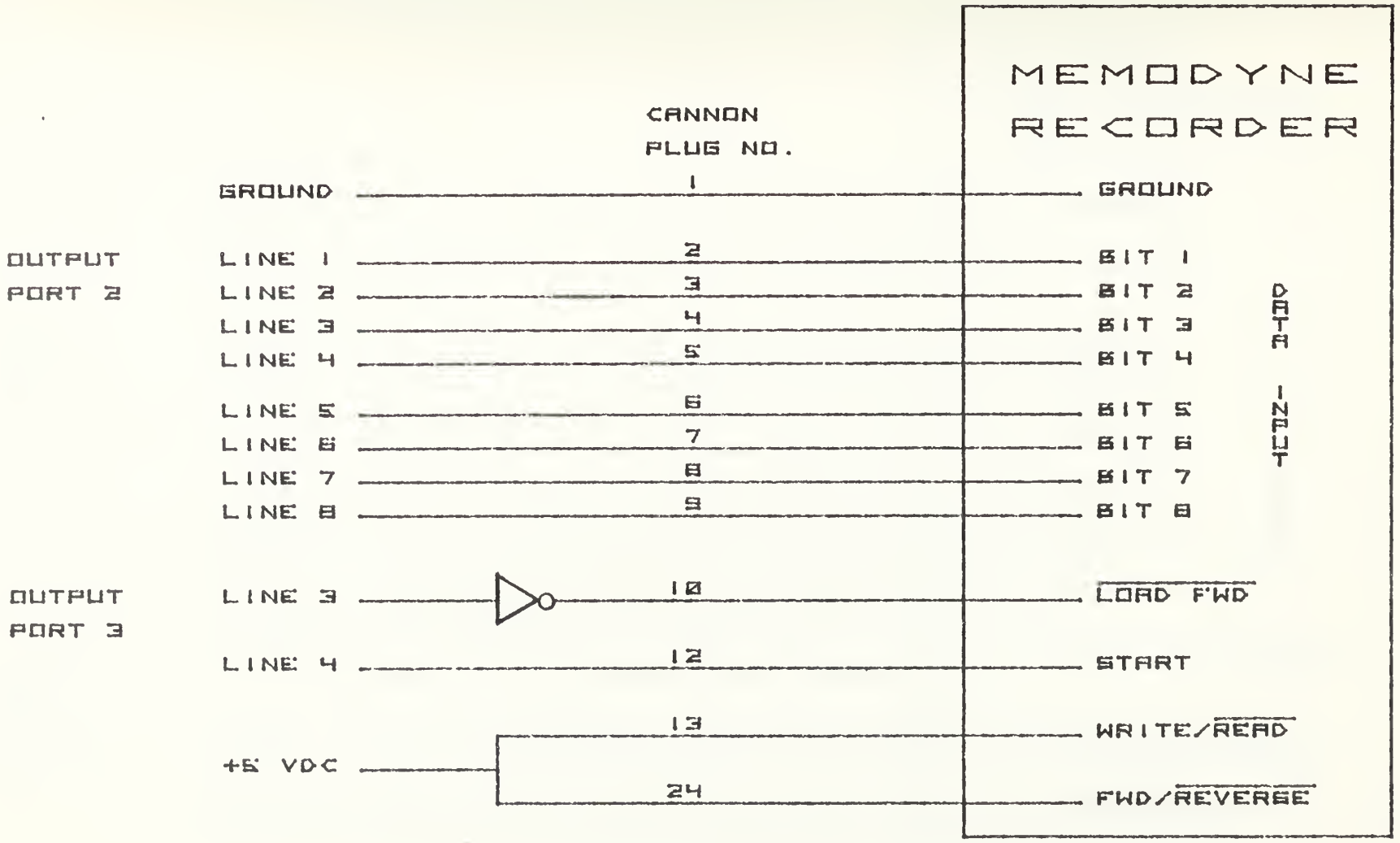

Figure 1 - Schematic: Memodyne Recorder Connections

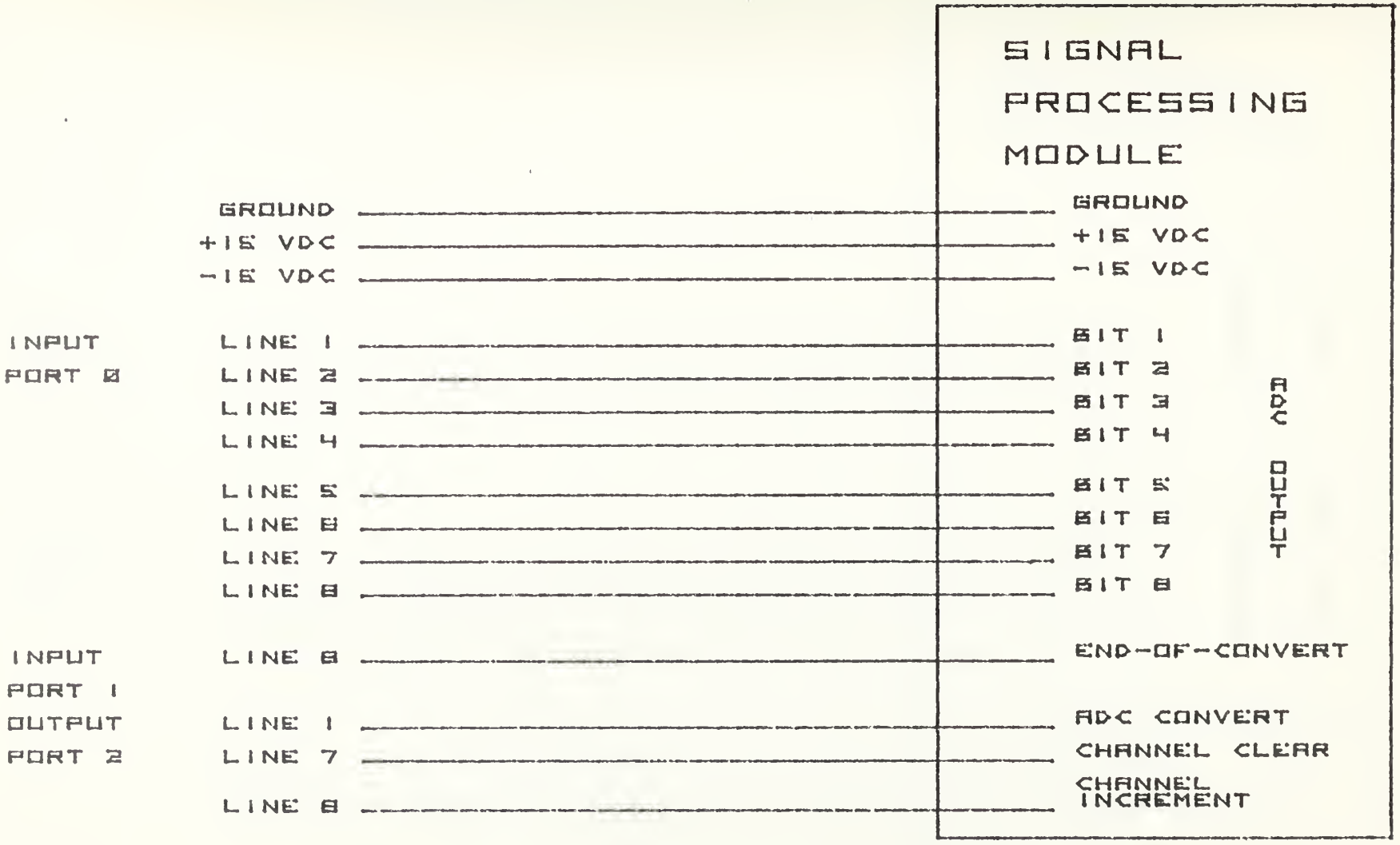

Figure 2 - Schematic: Signal Processing Module Connections

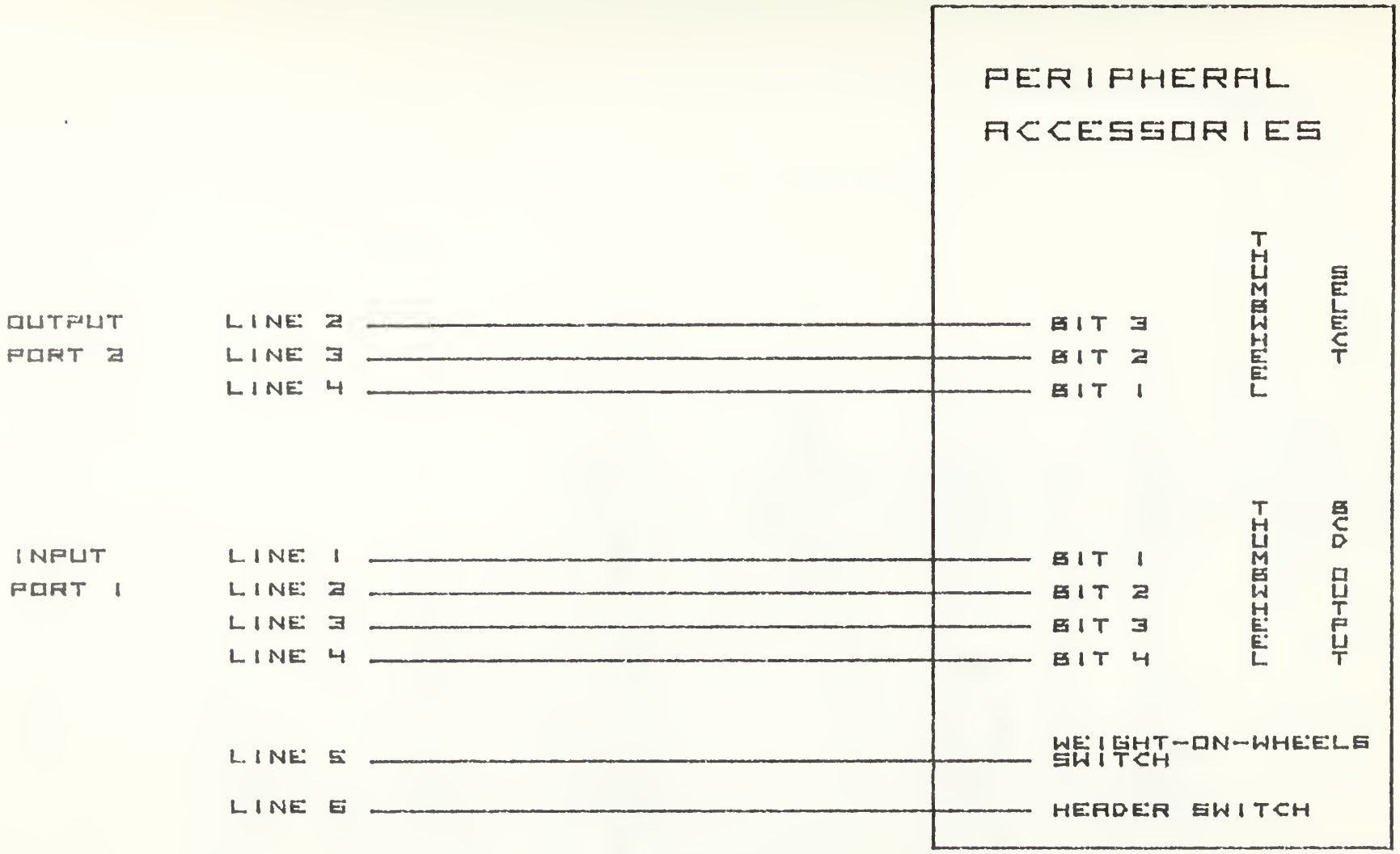

Figure 3 - Schematic: Peripheral Accessories Connections

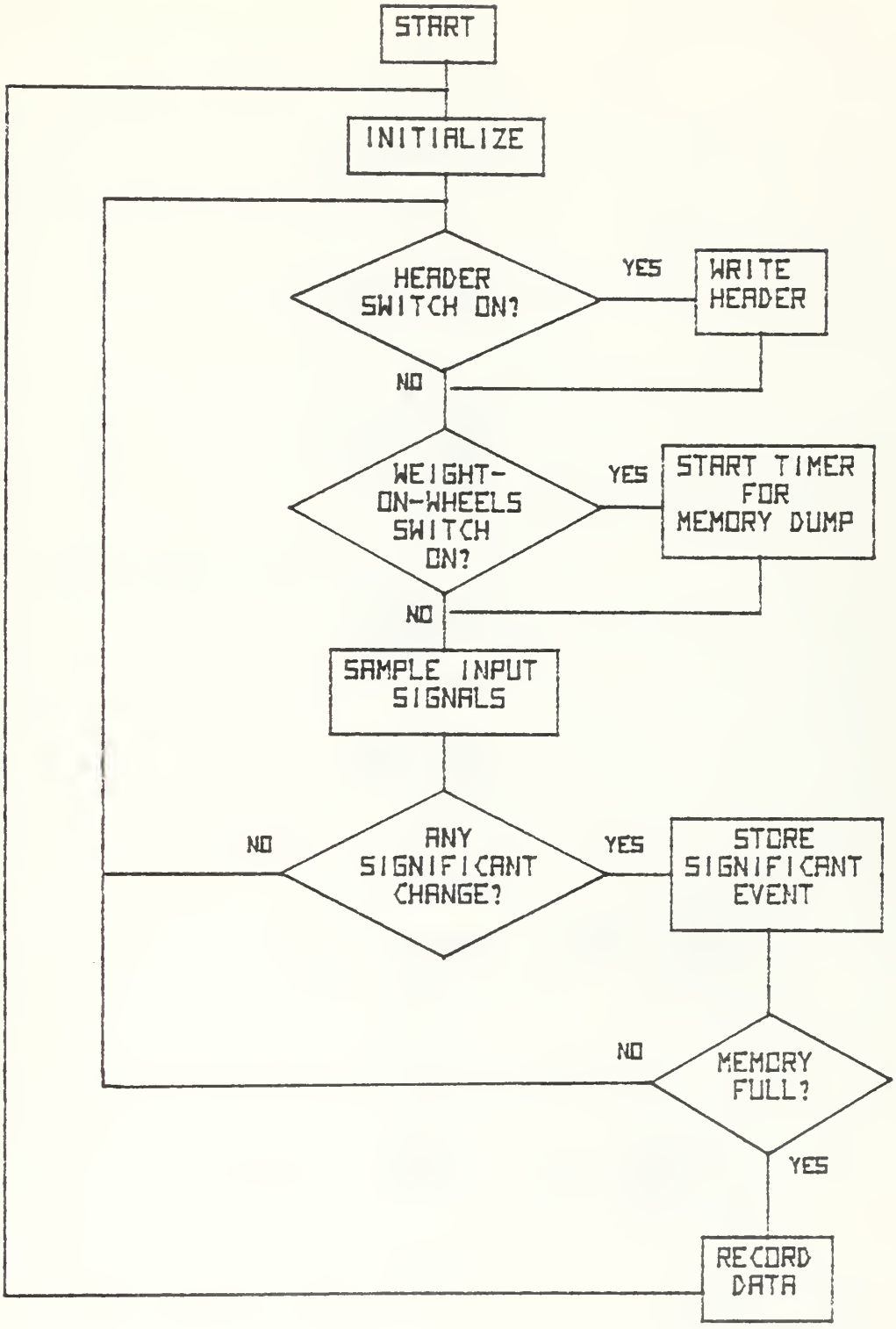

Figure 4 - Flowchart: WRITE Program

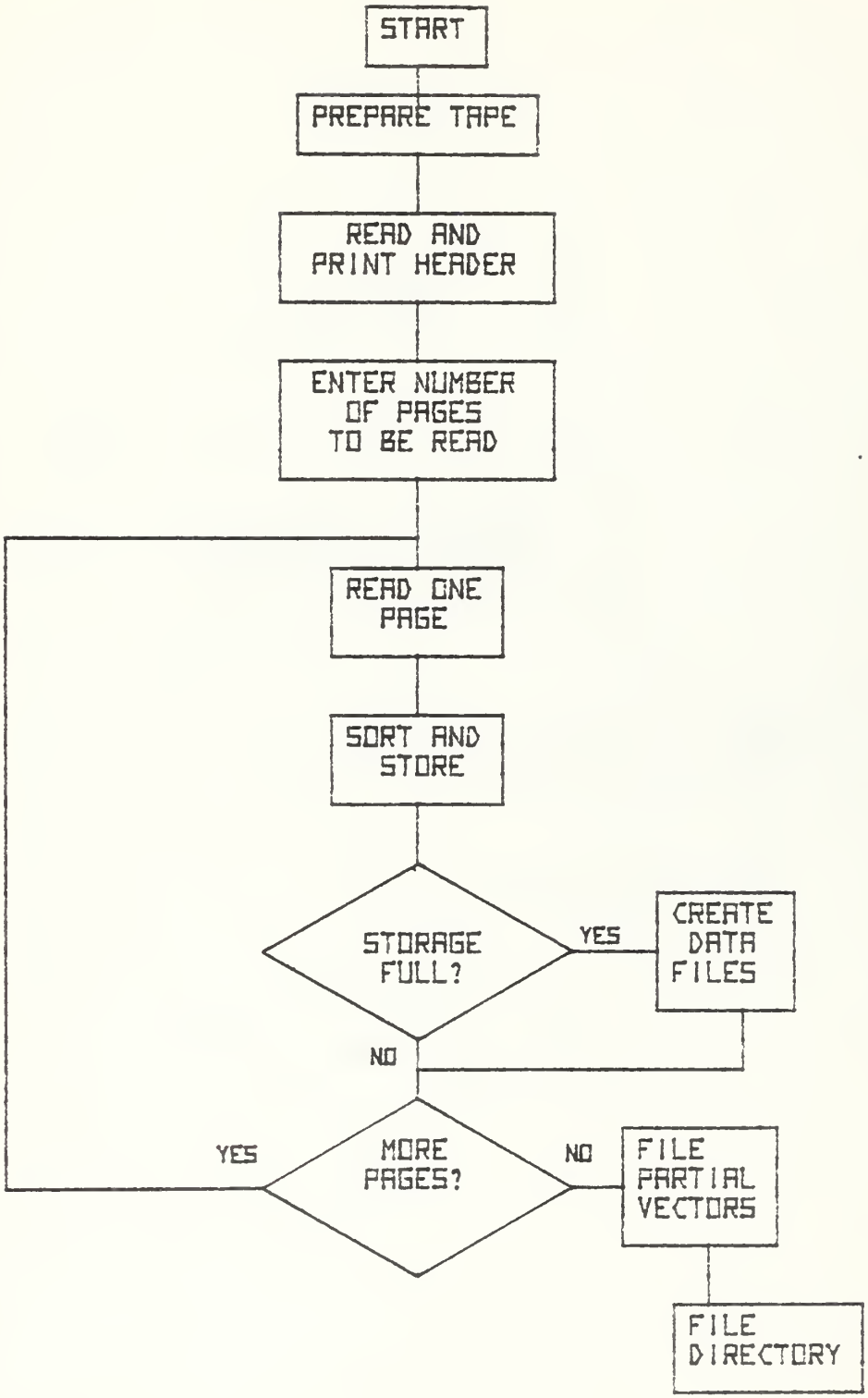

Figure 5 - Flowchart: READ/STORE Program

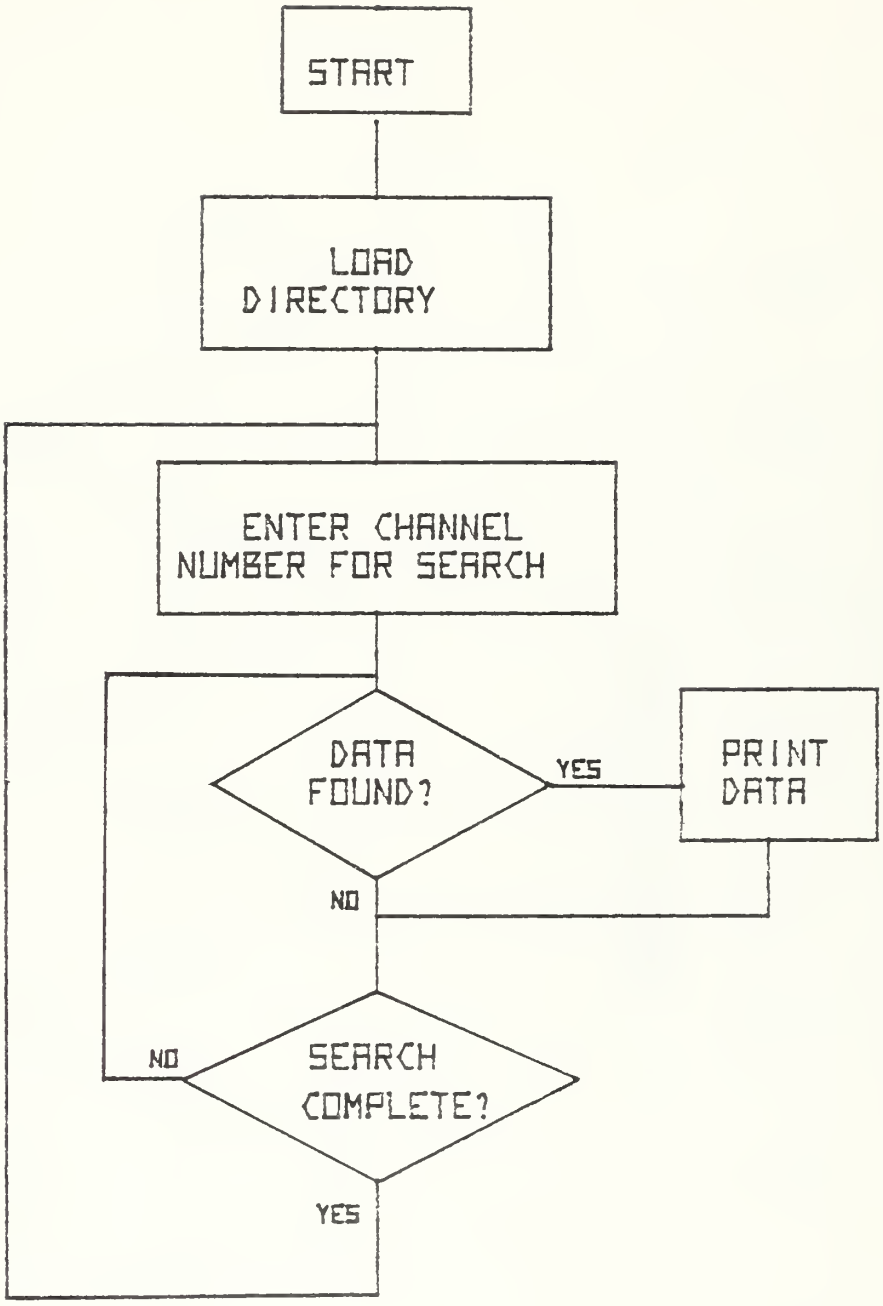

Figure 6 - Flowchart: File Retrieval Program

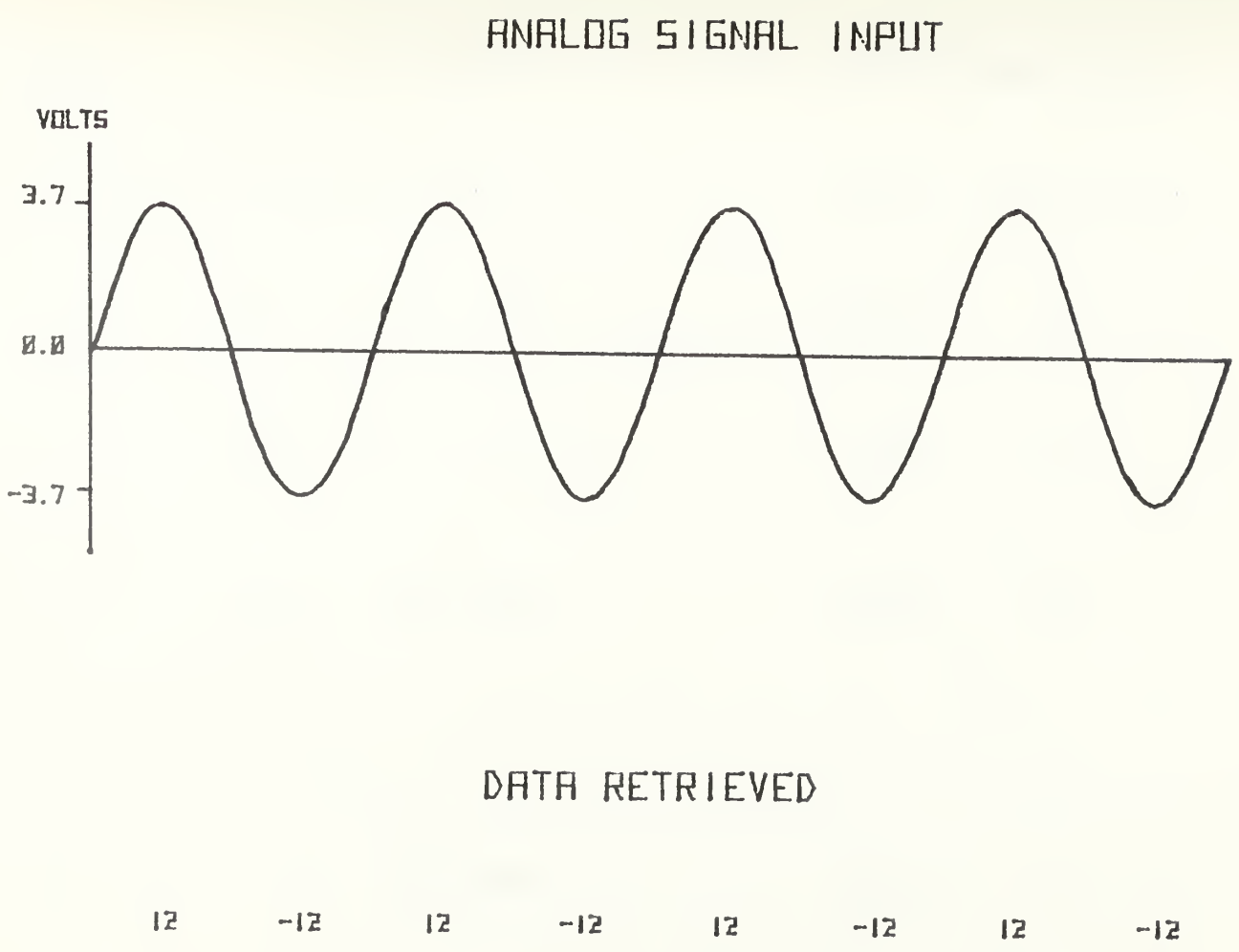

Figure 7 - Input/Output Comparison (Channel 2)

ANALOG SCALE	DIGITAL SCALE			5-BIT ROUNDED SCALE		READ SYSTEM SHIFTED SCALE
(VOLTS)	(DECIMAL)	(BINARY)		(BINARY)	(DECIMAL)	
5	256	10000 0000		100000	32	16
		(1111 1111)	UPPER BOUND	(11111)	(31)	(15)
3.7	223	1101 1111		11100	28	12
0	128	1000 0000		10000	16	0
-5	0	0000 0000		00000	0	-16

Figure 8 - Signal Scales

APPENDIX A

THE WRITE SYSTEM SOFTWARE PACKAGE

```
/*DECLARATIONS*/

DECLARE DCL LITERALLY 'DECLARE';
DCL LIT LITERALLY 'LITERALLY';
DCL (I,J,K) BYTE;
DCL (XX,LASTX,SIG,CHANNEL) BYTE;
DCL X(8) BYTE;
DCL XLST(8) BYTE;
DCL SLST(8) BYTE;
DCL FLAG(8) BYTE;
DCL C BYTE;
DCL RAM$LOC ADDRESS;
DCL EVENT BASED RAM$LOC BYTE;
DCL POS$THRESH LIT '14H';
DCL NEG$THRESH LIT '0DH';
DCL FOREVER LIT 'WHILE 0FFH';
DCL TIMER ADDRESS;
```

```
/*PROCEDURES*/

DELAY:PROCEDURE(VAL); /*PROVIDES TIME DELAY*/
  DCL VAL BYTE;
  CALL TIME(VAL);
END DELAY;

BYTE$RECORD:PROCEDURE; /*TRANSFERS ONE BYTE TO RECORDER*/
  OUTPUT(2)=EVENT;
  OUTPUT(3)=08H;
  OUTPUT(3)=0;
  CALL DELAY(77H);
END BYTE$RECORD;

GAP:PROCEDURE(N); /*WRITES BLANK SEGMENTS OF TAPE*/
  DCL N BYTE;
  OUTPUT(3)=04H;
  CALL DELAY(N);
  OUTPUT(3)=0;
END GAP;

RECORD:PROCEDURE; /*RECORDS A BLOCK OF DATA*/
  DCL COUNT ADDRESS;
  COUNT=RAM$LOC-1;
  DO RAM$LOC=0900H TO COUNT;
    CALL BYTE$RECORD;
  END;
  CALL GAP(80H);
  RAM$LOC=08FFH;
END RECORD;
```

```
STORE:PROCEDURE; /*STORES A BYTE IN RAM*/
  EVENT=NOT(LASTX OR ROR(CHANNEL,3)); /*PACKS WORD*/
  FLAG(CHANNEL)=((LASTX>POS$THRESH)
                 OR (LASTX<NEG$THRESH));
  RAM$LOC=RAM$LOC+1;
  IF RAM$LOC>=1000H THEN CALL RECORD; /*MEMORY FULL*/
END STORE;

READ:PROCEDURE BYTE; /*READS A THUMBWHEEL SWITCH*/
  OUTPUT(2)=C;
  C=C+2;
  RETURN (INPUT(1) AND 0FH) XOR 0FH;
END READ;

HEADER:PROCEDURE; /*PREPARES TAPE FOR DATA RECORDING*/
  DO K=1 TO 3;
    DO J=1 TO 150;
      CALL GAP(255); /*ADVANCES TAPE BEYOND LEADER*/
    END;
  END;

  DO WHILE ROL(INPUT(1),3); /*WAIT FOR HEADER SWITCH OFF*/
  END;
  C=0;
  DO RAM$LOC=0900H TO 0904H;
    EVENT=80H;
  END;
  DO RAM$LOC=0905H TO 0907H;
    EVENT=NOT(SHL(READ,4)+READ);
  END;
  RAM$LOC=0908H;
  CALL RECORD;
  RAM$LOC=0900H;
END HEADER;
```

```
MUX:PROCEDURE; /*SEQUENTIALLY MONITORS CHANNELS*/
  DCL L BYTE;
  OUTPUT(2)=40H;
  OUTPUT(2)=0;
  DO I=0 TO 7;
    OUTPUT(2)=1;
    OUTPUT(2)=0;
    DO WHILE NOT(ROL(INPUT(1),1));
    END;
    K=INPUT(0);
    OUTPUT(2)=80H;
    OUTPUT(2)=0;
    L=K AND 04H;
    L=SHL(L,1);
    K=K+L;
    X(I)=SHR(K,3);
  END;
END MUX;

INITIALIZE:PROCEDURE; /*INITIALIZES SYSTEM*/
  RAM$LOC=0900H;
  TIMER=0;
  CALL MUX;
  DO CHANNEL=0 TO 7;
    XLST(CHANNEL)=X(CHANNEL);
    SLST(CHANNEL)=0;
    FLAG(CHANNEL)=0;
  END;
END INITIALIZE;
```

```
DUMP:PROCEDURE; /*RECORDS MEMORY RESIDUE*/
  TIMER=TIMER+1;
  IF TIMER<0700H THEN RETURN;
  CALL RECORD;
  CALL INITIALIZE;
END DUMP;
```

```
/*MAIN PROGRAM*/

CALL INITIALIZE;

DO FOREVER;
  IF ROL(INPUT(1),3) THEN CALL HEADER;
                     /*CHECK HEADER SWITCH*/
  IF ROL(INPUT(1),4) THEN CALL DUMP;
                     /*CHECK WEIGHT-ON-WHEELS SWITCH*/
    ELSE TIMER=0;

  CALL MUX;
  DO CHANNEL=0 TO 7;
    IF (XX:=X(CHANNEL))<>(LASTX:=XLST(CHANNEL)) THEN
      DO;
        IF XX>LASTX THEN SIG=0FFH;
          ELSE SIG=0;
        IF ((((LASTX>POS$THRESH) OR (LASTX<NEG$THRESH))
          OR FLAG(CHANNEL))
          AND (SIG XOR SLST(CHANNEL))) THEN CALL STORE;
              /*DETERMINE SIGNIFICANCE*/
        XLST(CHANNEL)=XX;
        SLST(CHANNEL)=SIG;
      END;
    IF RAM$LOC=08FFH THEN CALL INITIALIZE;
  END;
END;

EOF
```

APPENDIX B

PL/M

PL/M is a block-structured language, which gives the program a modular appearance. The program has three obvious divisions: a series of declarations, a series of procedures, and the main program. The declarations are of two basic types: variable declarations and literal declarations. The variable declarations identify variable names to be used and the amount of memory storage each will require. The literal declarations are definitions assigned by the programmer; thus allowing the use of more descriptive program statements or shorthand notation for lengthy PL/M constructs.

The procedures are individual blocks of code, which perform specialized functions and have names somewhat descriptive of that function. A procedure is invoked by a CALL <procedure name> statement from elsewhere in the program. After a procedure has performed its function, the execution of the program continues at the point from which the procedure was called.

The main program is the starting point for the execution of the code. The main program is a series of executable steps, which performs various functions and invokes procedures to accomplish the program task.

PL/M variables are either of type BYTE or type ADDRESS. BYTE variables are eight binary digits in length,

while ADDRESS variables are sixteen bits long. All arithmetic or boolean operations which are performed are best understood if the binary representation of numbers is visualized. Hexadecimal numbers are easily converted to binary and are frequently used in PL/M programming. Hexadecimal numbers are distinguished by the letter H following the number. Binary numbers are identified by the suffix B and decimal numbers have no suffix. The dollar sign has no significance in PL/M and may be used to separate characters where a blank space would be inappropriate. For example, the expression 0100$0100B is equivalent to 01000100B. Use of the dollar sign frequently aids in the readability of the program.

The OUTPUT statement is of the form OUTPUT(<port number>)=<expression>. The expression represents an 8-bit pattern which corresponds to voltage levels to be latched in the output port whose number is indicated. For example,

```
OUTPUT(2)=40H;
```

is an instruction to latch voltage levels on lines corresponding to output port two in the pattern 0100$0000B (40H=0100$0000B) where 1 indicates a HIGH voltage and 0 indicates a LOW voltage. The eight lines of output port two have a one-to-one correspondence with the bit positions. The leftmost bit corresponds to line 8 and the rightmost bit corresponds to line 1. In the example only line 7 would have a HIGH voltage.

The use of INPUT instructions is similar. For example,

```
X=INPUT(0);
```

causes input port zero to be sampled and assigned to X as an 8-bit binary number. Each bit in the number would indicate the status of the line to which it corresponds. If in the example, X is found to be equal to 0001$0010B, then lines

two and five of input port zero have been determined to be the only lines at HIGH voltage.

Most examinations of bit patterns in PL/M require some bit manipulation because logical tests of numbers involve only the rightmost bit. A shift causes a bit pattern to be offset in the direction indicated with vacated positions being filled with zero. For example,

```
SHR(0111$1011B,3)
```

causes the bit pattern to be shifted to the right by three bit positions with the vacated positions being filled with zeroes. The result would be 0000$1111B. The three rightmost bits are lost. The shift operation may also be performed to the left.

The rotate operation is similar to the shift operation except that the vacated positions are filled with the displaced bits from the opposite end. For example,

```
ROL(1101$1000B,3)
```

causes the bit pattern to be rotated left by three positions. The result is 1100$0110B where the three bits displaced at the left appear in the three positions which were vacated at the right. The use of the shift and rotate constructs allow any bit to be selectively placed in the rightmost position. The rotate operation may also be performed to the right.

The IF <expression> and DO WHILE <expression> constructs require that the expression have a 1 in the rightmost bit position in order for the test condition to be satisfied. For example,

```
IF ROL(INPUT(1),3) THEN CALL HEADER;
```

will result in a procedure call only if line 6 of input port one is examined and found at HIGH status. The use of NOT before the expression would result in the procedure call being invoked only if line 6 were found to be at LOW status.

The DO WHILE <expression> causes the current block of code to be repetitively executed while the rightmost bit of the expression is a 1. The use of NOT before the expression causes the execution to continue while the rightmost bit is zero. For example,

```
DO WHILE NOT (ROL(INPUT(1),1));
X=X+1;
END;
```

causes X to be incremented until line 8 of input port one is HIGH.

The AND, OR, and XOR operators perform the boolean operations "and", "inclusive or", and "exclusive or" respectively bit-by-bit on their arguments. For example,

```
0111$0100B AND 1111$0000B
```

results in 0111$0000B. This operation is frequently used to isolate or mask a portion of the bit pattern.

APPENDIX C

THE READ SYSTEM SOFTWARE PACKAGE

The READ/STORE Program

```
100 DIM A[100],B[100],C[100],D[100]
110 DIM E[100],F[100],G[100],H[100]
120 DIM L[100],Y[100],Z[256]
130 MAT Y=CON
140 MAT Y=(100)*Y
150 MAT A=Y
160 MAT B=Y
170 MAT C=Y
180 MAT D=Y
190 MAT E=Y
200 MAT F=Y
210 MAT G=Y
220 MAT H=Y
230 MAT Z=ZER
240 MAT L=ZER
250 PRINT "ADVANCE TAPE BEYOND CLEAR LEADER"
260 PRINT "BY HOLDING 'STOP' KEY DEPRESSED"
270 PRINT "THEN PRESS 'CONT' 'EXECUTE'"LIN2
280 STOP
290 REM*****************************************************
400 DISP "ENTER DA,MO,YR",SPA20;
410 INPUT N1,N2,N3
420 FOR N=1 TO 8
430 Z[N]=RBYTE7
440 NEXT N
```

```
450 M0=0
460 FOR N=6 TO 8
470 M0=100*M0
480 M1=BIAND(Z[N],240)
490 M1=ROT(M1,4)
500 M1=10*M1
510 M0=M0+M1
520 M1=BIAND(Z[N],15)
530 M0=M0+M1
540 NEXT N
550 PRINT "BUNO=";M0,"DA/MO/YR=";N1,N2,N3
560 REM*****************************************************
1000 N=N1=N2=N3=N4=N5=N6=N7=N8=1
1010 M0=M1=2
1020 DISP "ENTER NO. OF PAGES TO BE READ"SPA5;
1030 INPUT P
1040 FOR K=1 TO P
1050 FOR I=1 TO 256
1060 Z[I]=RBYTE7
1070 NEXT I
1080 FOR I=1 TO 256
1090 J=BIAND(Z[I],224)
1100 J=ROT(J,5)
1110 J=J+1
1120 V=BIAND(Z[I],31)
1130 V=V-16
1140 GOSUB J OF 4100,4200,4300,4400,4500,4600,4700,4800
1150 NEXT I
1160 NEXT K
1170 REM*****************************************************
2000 V=100
2010 N1=N2=N3=N4=N5=N6=N7=N8=100
2020 FOR J=1 TO 8
2030 GOSUB J OF 4100,4200,4300,4400,4500,4600,4700,4800
2040 NEXT J
2050 STORE  DATA M0,L
```

```
2060 REWIND
2070 STOP
2080 REM*************************************************
4100 A[N1]=V
4110 N1=N1+1
4120 IF N1≠101 THEN 4199
4140 GOSUB 7000
4150 STORE  DATA M1,A
4180 N1=1
4190 MAT A=Y
4199 RETURN
4200 B[N2]=V
4210 N2=N2+1
4220 IF N2≠101 THEN 4299
4240 GOSUB 7000
4250 STORE  DATA M1,B
4280 N2=1
4290 MAT B=Y
4299 RETURN
4300 C[N3]=V
4310 N3=N3+1
4320 IF N3≠101 THEN 4399
4340 GOSUB 7000
4350 STORE  DATA M1,C
4380 N3=1
4390 MAT C=Y
4399 RETURN
4400 D[N4]=V
4410 N4=N4+1
4420 IF N4≠101 THEN 4499
4440 GOSUB 7000
4450 STORE  DATA M1,D
4480 N4=1
4490 MAT D=Y
4499 RETURN
4500 E[N5]=V
```

```
4510 N5=N5+1
4520 IF N5≠101 THEN 4599
4540 GOSUB 7000
4550 STORE  DATA M1,E
4580 N5=1
4590 MAT E=Y
4599 RETURN
4600 F[N6]=V
4610 N6=N6+1
4620 IF N6≠101 THEN 4699
4640 GOSUB 7000
4650 STORE  DATA M1,F
4680 N6=1
4690 MAT F=Y
4699 RETURN
4700 G[N7]=V
4710 N7=N7+1
4720 IF N7≠101 THEN 4799
4740 GOSUB 7000
4750 STORE  DATA M1,G
4780 N7=1
4790 MAT G=Y
4799 RETURN
4800 H[N8]=V
4810 N8=N8+1
4820 IF N8≠101 THEN 4899
4840 GOSUB 7000
4850 STORE  DATA M1,H
4880 N8=1
4890 MAT H=Y
4899 RETURN
4900 REM**************************************************
7000 L[N]=J
7010 N=N+1
7020 M1=M1+1
7030 RETURN
```

The File Retrieval Program

```
10 DIM L[100],X[100]
20 M=2
30 LOAD DATA M,L
35 PRINT LIN2
40 DISP "ENTER CHANNEL NO.(1-8)";
50 INPUT C
60 PRINT "CHANNEL";C,LIN1
70 FOR N=1 TO 100
80 IF L[N]≠C THEN 90
85 GOSUB 120
90 IF L[N]=0 THEN 35
95 NEXT N
99 REM**************************************************
120 LOAD DATA N+M,X
130 FOR I=1 TO 91 STEP 10
140 PRINT X[I];X[I+1];X[I+2];X[I+3];X[I+4];X[I+5];X[I+6];
          X[I+7];X[I+8];X[I+9]
150 NEXT I
160 PRINT
170 RETURN
```

APPENDIX D

BASIC

The statements of a program written in BASIC are executed sequentially in the numerical order of their statement numbers. The storage of variables is allocated upon initial use of the variable with the exception of arrays. Array storage must be declared in advance by the use of the DIM (dimension) statement.

The main program follows the DIM statements and any initialization statements which might be required. The main program is followed by the procedural subroutines. A subroutine is delimited by its beginning statement number and an ending RETURN statement. A subroutine is invoked by a GOSUB <beginning statement number> statement from elsewhere in the program. The RETURN statement causes the program to resume with the next statement following the GOSUB statement.

All numbers appearing in BASIC are decimal numbers. Bit manipulative operations return the decimal representation of the 8-bit binary result. The boolean "and" is expressed as

```
X=BIAND(A,B)
```

and causes X to be assigned the decimal value of the binary result of the bit-by-bit "and"-ing of the binary representation of A with the binary representation of B. For example,

```
X=BIAND(14,9)
```

is equivalent to X=0000$1110B AND 0000$1001B=0000$1000B and would result in X=8. Similarly the ROT statement is used for right rotation. The statement

```
X=ROT(A,B)
```

causes X to be assigned the decimal value of the binary result of the rotation of the bit pattern of A to the right for B positions. For example,

```
X=ROT(17,3)
```

is equivalent to the rotation of 0001$0001B to the right for three positions which results in X=0010$0010B=34.

The prefix MAT identifies a statement as a matrix operation. The statement

```
MAT A=ZER
```

causes all elements of the array A to be set equal to zero. The statement

```
MAT B=CON
```

causes all elements of array B to be set equal to one. The statement

```
MAT X=Y
```

causes each element of array X to be set equal to the corresponding element of array Y.

The PRINT statement is used to output information on the peripheral line printer. The display command (DISP) causes a line to be output in the display window above the keyboard. Blank spaces or lines may be indicated by using SPA or LIN followed by a number.

The INPUT command interrupts execution of the program and awaits the input of data by the user at the keyboard. The statement

```
INPUT A,B
```

causes anticipation of a user response. For example, a user response of

```
4,6.5 <EXECUTE>
```

assigns values of 4 and 6.5 to variables A and B respectively before continuing program execution.

The conditional IF <argument> THEN <statement number> causes the program execution to be diverted to the statement number following THEN if the argument following IF is a true statement. For example,

```
IF X<6 THEN 100
```

results in program branching to statement 100 only if the variable X is currently less than 6.

Another powerful conditional branching statement is the GOSUB <expression> OF <statement number list> statement. The value of the expression is an ordinal number, which causes the execution of the program to branch to the statement number appearing in the corresponding position in the list of statement numbers. For example,

```
X=3
GOSUB X OF 100,150,200,250
```

would cause the subroutine located at line 200 to be executed.

LIST OF REFERENCES

1. Biewer, M., The Designer's Guide to Programmed Logic, Pro-Log Corporation, 1974.

2. Intel Corporation, 8008 and 8080 PL/M Programming Manual, 1975.

3. Stanfield, W.C., Microprogrammable Integrated Data Acquisition System: Fatigue Life Data Application, MSAE Thesis, Naval Postgraduate School, Monterey, California, 1976.

4. Naval Air Development Center Report 13920-1, Aircraft Structural Appraisal of Fatigue Effects (SAFE) Program, by Rudolph R. Virga, 15 July 1976.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 67 Department of Aeronautics Naval Postgraduate School Monterey, California 93940	1
4.	Assoc. Professor G.H. Lindsey, Code 67Li Department of Aeronautics Naval Postgraduate School Monterey, California 93940	1
5.	Lt. C.L. Butler, U.S.N. FLECOMPRON ONE F.P.O. San Francisco, California 96601	1
6.	Dr. David E. Weiss, Code 3033 Aero Structures Department Naval Air Development Center Warminster,Pennsylvania 18974	1
7.	Dr. S.L. Huang, Code 3033 Aero Structures Department Naval Air Development Center Warminster,Pennsylvania 18974	1

8. Dr. A.E. Sommeroff, Code 320 1
Naval Air Systems Command
Jefferson Plaza No. 1
Washington, D.C. 20361

Thesis
B935 Butler
c.1 Software design for
a fatigue monitoring
data acquisition sys-
tem.
166559